I can't get no satisfaction...

But the proof is exponential

So I guess I should be happy

LAURENT SIMON
BORDEAUX-INP / LABRI
UNIVERSITY OF BORDEAUX

@lorensipro
lsimon@labri.fr
Playing with lights
Playing with lights

v1

v2

v3
Playing with lights
Playing with lights

v1

v2

v3
Playing with (a few) more lights

Already hard to guess with 10 variables and 15 lights...
What about millions of them?
Playing with (a few) more lights

Already hard to guess with 10 variables and 15 lights... What about millions of them?
Playing with (a few) more lights

Already hard to guess with 10 variables and 15 lights...
What about millions of them?
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The facts:
are true, or false.

The variables:
are $T$, or $\bot$.

Reasoning by calculus

If we know that:

$\begin{align*}
\Rightarrow & A \text{ implies } B \\
\Rightarrow & B \text{ implies } C
\end{align*}$

Then, we can deduce:

$\Rightarrow A \text{ implies } C$

If we have:

$\begin{align*}
\Rightarrow & \neg A \lor B \\
\Rightarrow & \neg B \lor C
\end{align*}$

Then, by resolution:

$\Rightarrow \neg A \lor C$

Logic defined more than 2300 years ago!
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The facts:
are true, or false.

The variables:
are $T$, or $\bot$.

Reasoning by calculus

If we know that:

$> A \implies B$

$> B \implies C$

Then, we can deduce:

$> A \implies C$

If we have:

$> \neg A \lor B$

$> \neg B \lor C$

Then, by resolution:

$> \neg A \lor C$

Logic defined more than 2300 years ago!
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The facts: are true, or false.

The variables: are $T$, or $\bot$.

Reasoning by calculus

If we know that:

- $A$ implies $B$
- $B$ implies $C$

Then, we can deduce:

- $A$ implies $C$

If we have:

- $\neg A \lor B$
- $\neg B \lor C$

Then, by resolution:

- $\neg A \lor C$

Logic defined more than 2300 years ago!
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The facts: are true, or false.
The variables: are \( T \), or \( \bot \).

Reasoning by calculus

If we know that:
> A implies B
> B implies C

Then, we can deduce:
> A implies C

If we have:
> \( \neg A \lor B \)
> \( \neg B \lor C \)

Then, by resolution:
> \( \neg A \lor C \)

Logic defined more than 2300 years ago!
Propositional Logic: « Simple is beautiful »

Knowledge Representation

The facts: are true, or false. The variables: are $T$, or $\bot$.

Reasoning by calculus

If we know that:
- $A$ implies $B$
- $B$ implies $C$

Then, we can deduce:
- $A$ implies $C$

If we have:
- $\neg A \lor B$
- $\neg B \lor C$

Then, by resolution:
- $\neg A \lor C$

Logic defined more than 2300 years ago!
Lights on Clauses and Variables

Translation:
\[
(\neg v_4 \lor \neg v_1 \lor v_9) \land (v_9 \lor \neg v_8 \lor v_7) \land (\neg v_6 \lor \neg v_{10} \lor v_5) \land (\neg v_3 \lor \neg v_{10} \lor \neg v_4) \land (\neg v_{10} \lor \neg v_2 \lor v_9) \land (v_4 \lor \neg v_8 \lor \neg v_9) \land (\neg v_2 \lor v_6 \lor \neg v_7) \land (v_5 \lor \neg v_6 \lor v_1) \land (v_2 \lor \neg v_6 \lor \neg v_4) \land (\neg v_3 \lor \neg v_9 \lor v_7) \land (\neg v_2 \lor \neg v_7 \lor v_3) \land (v_1 \lor v_2 \lor v_9) \land (v_2 \lor \neg v_{10} \lor v_4) \land (\neg v_{10} \lor v_1 \lor v_2) \land (\neg v_8 \lor v_4 \lor v_9)
\]

Each lamp is a clause. Each switch is a propositional variable.
Each lamp is a clause. Each switch is a propositional variable.
Introduction
(again)
Why studying SAT?

SAT, the canonical NP-Complete problem.

Related to a one-million dollar question (is $\text{NP}=\text{P}$?)

- The main open problem of Theoretical Computer Science
- Must be faced in most real-world problems
- But, the easiest of the hard problems

$S. \ Aaronson, \ MIT:$ « If $\text{P} = \text{NP}$, then the world would be a profoundly different place than we usually assume it to be. There would be no special value in ‘creative leaps,’ no fundamental gap between solving a problem and recognizing the solution once it’s found. Everyone who could appreciate a symphony would be Mozart; everyone who could follow a step-by-step argument would be Gauss. »

\(^1\) citation taken from [Vardi, 2015]
Why studying SAT? – 2

Equivalence to SAT: **prove the hardness of new problems**
- many reductions exist from many problems to SAT
- SAT "captures" the difficulty of many other problems

**SAT Solvers are (incredibly) efficient and "user-friendly"**

- Can be used as a **black box**
- Can be used as an **open box** (special heuristics, ...)
- Can be **extended** in many ways
  - SAT solvers as Oracles on many close formulas
  - SAT solvers working at an abstract level
  - SAT solvers working on "on the fly" generated constraints
  - ...

---

Rennes 28-11-2019
Why studying SAT? – 3

D. Knuth (volume 4, Fascicle 6):
"The story of satisfiability is the tale of a triumph of software engineering, blended with rich doses of beautiful mathematics. Thanks to elegant new data structures and other techniques, modern SAT solvers are able to deal routinely with practical problems that involve many thousands of variables, although such problems were regarded as hopeless just a few years ago."

E. Clarke:
"The practical solving of SAT is a key technology for computer science in the 21st century."

Everybody: "And it is A.I.!!!"
Why studying SAT? – 3

D. Knuth (volume 4, Fascicle 6):
"The story of satisfiability is the tale of a triumph of software engineering, blended with rich doses of beautiful mathematics. Thanks to elegant new data structures and other techniques, modern SAT solvers are able to deal routinely with practical problems that involve many thousands of variables, although such problems were regarded as hopeless just a few years ago."

E. Clarke:
"The practical solving of SAT is a key technology for computer science in the 21st century."

Everybody: "And it is A.I.!!!"
Simple is beautiful... But speed is essential

The paradigm shift is essentially due to

- Tight data structures
- Algorithms built upon this data structure

The way "Modern" SAT solvers are solving problems has nothing common with a human strategy

Interesting problems are not toy problems

Beeing fast is the way computers are not so dumb
The SAT problem

Definition

**Input**: A set of clauses built from a propositional language with \( n \) variables.

**Output**: Is there an assignment of the \( n \) variables that satisfies all those clauses?

**Example** (with different notations)

\[
\Sigma_1 = (\neg a \lor b) \land (\neg b \lor c) = (a' + b). (b' + c) = \{ \neg a \lor b, \neg b \lor c \}
\]

\[
\Sigma_2 = \Sigma_1 \land a \land \neg c = \Sigma_1 \cup \{ a, \neg c \}
\]

For \( \Sigma_1 \), the answer is yes, for \( \Sigma_2 \) the answer is no

\[
\Sigma_1 \models \neg a \lor c = \neg (a \land \neg c)
\]
Why working on CNF? Bec(l)ause!

Why considering SAT on CNF only?

(SAT is trivial on DNF!)

- Human-designed systems are conjunctions of properties (in general)
- We want to do symbolic reasoning (not trying all possible solutions)
- There are (too) many ways of applying rules at each step
- We need to restrict the possibilities at each step

But, what about rewriting any formula into CNF?

- Very hard in general
- But, very easy if we just want to check SAT
**Working on CNF** [Tseitin 1968]

**Idea:** introduce new variables encoding the satisfiability of subformulas

Let’s say we need to check $SAT(f)$ with

$$f \equiv g \lor h$$

We introduce $x_f$, $x_g$ and $x_h$ representing the satisfiability of $f$, $g$ and $h$, respectively

$$(\neg x_f \lor x_g \lor x_h) \land (x_f \lor \neg x_g) \land (x_f \lor \neg x_h)$$

$x_f$ encodes the satisfiability of $f$. Easy! (linear, no blow-up!)

*(Introducing new variables is so powerful, isn’t it?)*
It is like naming all the wires in a circuit

\[ f = (x_1 \land x_2) \lor ((x_3 \land x_4) \oplus (x_5 \land x_6)) \]

Adding \( y_1, y_2, y_3, y_4, y_f \) and:

\[ \Sigma_f \equiv \begin{cases} 
(y_1 \leftrightarrow x_1 \land x_2) \\
(y_2 \leftrightarrow x_3 \land x_4) \\
(y_3 \leftrightarrow x_5 \land x_6) \\
(y_4 \leftrightarrow y_2 \oplus y_3) \\
(y_f \leftrightarrow y_1 \land y_4) \\
\land (y_f) 
\end{cases} \]

\( f \) is satisfiable iff \( \Sigma_f \) is
Big (and even bigger) questions

Big questions

- **SAT**: is there an assignment of variables making the formula true?
- **UNSAT**: is the theory contradictory?
- **PI**: deduce all you can from $\Sigma$

How to solve SAT?

- Try all possible solutions?
- Try to guess a solution?
- Any other idea?
At the heart of most procedures: resolution

**The Resolution Rule (Cut)** [Gentzen 1934, Robinson 1965]

Let \( c_1 = (x \lor a_1 \lor \ldots a_n) \) and \( c_2 = (\neg x \lor b_1 \lor \ldots b_m) \)

\[
    c = (a_1 \lor \ldots a_n \lor b_1 \lor \ldots b_m)
\]

is obtained by resolution on \( x \) between \( c_1 \) and \( c_2 \)

It is a particular case of the following **deduction rule**:

\[
    \text{if } a \rightarrow b \text{ and } b \rightarrow c \text{ then } a \rightarrow c
\]

In general, SAT solvers are only using this rule
(but many, many times per second)

Finding which ones to trigger is the secret of efficient SAT solvers
Searching for solutions contradictions
Davis & Putnam: the firsts SAT steps

1958: Hilary Putnam and Martin Davis look for funding their research around propositional logic

« What we’re interested in is good algorithms for propositional calculus » (NSA)

Before that, only inefficient methods (truth tables, …)

First papers

> Computational Methods in The Propositional calculus
  [Davis Putnam 1958]\(^2\)

> A Computing Procedure for Quantification Theory
  [Davis Putnam 1960]

\(^2\)Rapport interne NSA
1960, already a first (kind of) competition!

« The superiority of the present procedure (i.e. DP) over those previously available is indicated in part by the fact that a formula on which Gilmore’s routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by hand computation using the present method in 30 minutes »

[Davis et Putnam 1960], page 202.

Every year since 2002, a new cycle of competitions, promoting:

- open source
- benchmarks collections
- fairness and fun
- tracks for students
1962-2001: DPLL rules the world

Systematically explore the space of partial models (backtrack)

> Choose a literal
> Try to find a solution with this literal set to True
> If it is not possible:
  Finds a solution with this literal set to False

Backtrack search on partial models
Systematic (ordered) exploration ensures completeness
1962-2001: DPLL rules the world

Systematically explore the space of partial models (backtrack)

> Choose a literal
> Try to find a solution with this literal set to True
> If it is not possible:
  Finds a solution with this literal set to False

Backtrack search on partial models
Systematic (ordered) exploration ensures completeness
Backtrack search

How to choose the right literal to branch on?
First search for a model or a contradiction?
Backtrack search

How to choose the right literal to branch on?
First search for a model or a contradiction?
An example of DPLL

Formula

\[
\begin{align*}
x_1 \lor x_4 \\
x_1 \lor x_4 \lor x_{14} \\
x_1 \lor \overline{x}_3 \lor \overline{x}_8 \\
x_1 \lor x_8 \lor x_{12} \\
x_2 \lor x_{12} \\
\overline{x}_3 \lor \overline{x}_{12} \lor x_{13} \\
x_3 \lor x_7 \lor \overline{x}_{13} \\
x_8 \lor \overline{x}_7 \lor \overline{x}_{12}
\end{align*}
\]

Simplified Formula

\[
\begin{align*}
x_1 \lor x_4 \\
\overline{x}_1 \lor x_4 \lor x_{14} \\
x_1 \lor \overline{x}_3 \lor \overline{x}_8 \\
x_1 \lor x_8 \lor x_{12} \\
x_2 \lor x_{12} \\
\overline{x}_3 \lor \overline{x}_{12} \lor x_{13} \\
\overline{x}_3 \lor x_7 \lor \overline{x}_{13} \\
\overline{x}_8 \lor \overline{x}_7 \lor \overline{x}_{12}
\end{align*}
\]

Partial Model

\[
\begin{align*}
\text{Lev. Lit. Back?}
\end{align*}
\]

\[x_1 \text{ appears in 4 clauses and 1 binary clause}\]
An example of DPLL

### Formula

\[
x_1 \lor x_4 \\
x_1 \lor \overline{x_4} \lor x_{14} \\
x_1 \lor x_3 \lor \overline{x_8} \\
x_1 \lor x_8 \lor x_{12} \\
x_2 \lor x_{12} \\
\overline{x_3} \lor \overline{x_{12}} \lor x_{13} \\
x_3 \lor x_7 \lor \overline{x_{13}} \\
x_8 \lor \overline{x_7} \lor \overline{x_{12}}
\]

### Simplified Formula

\[
x_1 \lor x_4 \\
\overline{x_1} \lor x_4 \lor x_{14} \\
x_1 \lor \overline{x_3} \lor \overline{x_8} \\
x_1 \lor x_8 \lor x_{12} \\
x_2 \lor x_{12} \\
\overline{x_3} \lor \overline{x_{12}} \lor x_{13} \\
\overline{x_3} \lor x_7 \lor \overline{x_{13}} \\
x_8 \lor \overline{x_7} \lor \overline{x_{12}}
\]

### Partial Model

\[
\begin{array}{c|c}
\text{Lev. Lit. Back?} \\
1 & \overline{x_1} \\
\end{array}
\]

\(x_4\) appears in 1 unary clause
An example of DPLL

**Formula**

\[
\begin{align*}
x_1 \lor x_4 \\
\overline{x_1} \lor x_4 \lor x_{14} \\
x_1 \lor \overline{x_3} \lor \overline{x_8} \\
x_1 \lor x_8 \lor x_{12} \\
x_2 \lor x_{12} \\
x_3 \lor x_{12} \lor x_{13} \\
\overline{x_3} \lor x_7 \lor \overline{x_{13}} \\
x_8 \lor \overline{x_7} \lor \overline{x_{12}}
\end{align*}
\]

**Simplified Formula**

\[
\begin{align*}
x_1 \lor x_4 \\
\overline{x_1} \lor x_4 \lor x_{14} \\
x_1 \lor \overline{x_3} \lor \overline{x_8} \\
x_1 \lor x_8 \lor x_{12} \\
x_2 \lor x_{12} \\
x_3 \lor x_{12} \lor x_{13} \\
\overline{x_3} \lor x_7 \lor \overline{x_{13}} \\
x_8 \lor \overline{x_7} \lor \overline{x_{12}}
\end{align*}
\]

**Partial Model**

\[
\begin{align*}
\text{Lev. Lit. Back?} \\
1 & \quad \overline{x_1} \\
& \quad (d) \\
+ & \quad x_4
\end{align*}
\]

\[x_3 \text{ appears in } 3 \text{ clauses incl. } 1 \text{ (new) binary clause}\]
**An example of DPLL**

<table>
<thead>
<tr>
<th>Formula</th>
<th>Simplified Formula</th>
<th>Partial Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 \lor x_4$</td>
<td>$x_1 \lor x_4$</td>
<td>Lev. Lit. Back?</td>
</tr>
<tr>
<td>$\overline{x}<em>1 \lor x_4 \lor x</em>{14}$</td>
<td>$\overline{x}<em>1 \lor x_4 \lor x</em>{14}$</td>
<td>1 \ $\overline{x}_1$ (d)</td>
</tr>
<tr>
<td>$x_1 \lor \overline{x}_3 \lor \overline{x}_8$</td>
<td>$x_1 \lor \overline{x}_3 \lor \overline{x}_8$</td>
<td>+ \ $x_4$</td>
</tr>
<tr>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td>2 \ $x_3$ (d)</td>
</tr>
<tr>
<td>$x_2 \lor x_{12}$</td>
<td>$x_2 \lor x_{12}$</td>
<td></td>
</tr>
<tr>
<td>$\overline{x}<em>3 \lor \overline{x}</em>{12} \lor x_{13}$</td>
<td>$\overline{x}<em>3 \lor \overline{x}</em>{12} \lor x_{13}$</td>
<td></td>
</tr>
<tr>
<td>$x_3 \lor x_7 \lor \overline{x}_{13}$</td>
<td>$x_3 \lor x_7 \lor \overline{x}_{13}$</td>
<td></td>
</tr>
<tr>
<td>$x_8 \lor \overline{x}<em>7 \lor \overline{x}</em>{12}$</td>
<td>$x_8 \lor \overline{x}<em>7 \lor \overline{x}</em>{12}$</td>
<td></td>
</tr>
</tbody>
</table>

$\overline{x}_8$ appears in one unary clause
An example of DPLL

**Formula**

\[ x_1 \lor x_4 \]
\[ x_1 \lor x_4 \lor x_{14} \]
\[ x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ x_1 \lor x_8 \lor x_{12} \]
\[ x_2 \lor x_{12} \]
\[ \overline{x}_3 \lor x_{12} \lor x_{13} \]
\[ x_3 \lor x_7 \lor \overline{x}_{13} \]
\[ x_8 \lor \overline{x}_7 \lor \overline{x}_{12} \]

**Simplified Formula**

\[ x_1 \lor x_4 \]
\[ x_1 \lor x_4 \lor x_{14} \]
\[ x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ x_1 \lor x_8 \lor x_{12} \]
\[ x_2 \lor x_{12} \]
\[ \overline{x}_3 \lor x_{12} \lor x_{13} \]
\[ x_3 \lor x_7 \lor \overline{x}_{13} \]
\[ x_8 \lor \overline{x}_7 \lor \overline{x}_{12} \]

**Partial Model**

<table>
<thead>
<tr>
<th>Lev.</th>
<th>Lit.</th>
<th>Back?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \overline{x}_1 )</td>
<td>(d)</td>
</tr>
<tr>
<td>+</td>
<td>( x_4 )</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( x_3 )</td>
<td>(d)</td>
</tr>
<tr>
<td>+</td>
<td>( \overline{x}_8 )</td>
<td></td>
</tr>
</tbody>
</table>

\( x_{12} \) appears in 1 unary clause
An example of DPLL

<table>
<thead>
<tr>
<th>Formula</th>
<th>Simplified Formula</th>
<th>Partial Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 \lor x_4$</td>
<td>$x_1 \lor x_4$</td>
<td></td>
</tr>
<tr>
<td>$\overline{x}<em>1 \lor x_4 \lor x</em>{14}$</td>
<td>$\overline{x}<em>1 \lor x_4 \lor x</em>{14}$</td>
<td></td>
</tr>
<tr>
<td>$x_1 \lor \overline{x}_3 \lor \overline{x}_8$</td>
<td>$x_1 \lor \overline{x}_3 \lor \overline{x}_8$</td>
<td></td>
</tr>
<tr>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td></td>
</tr>
<tr>
<td>$x_2 \lor x_{12}$</td>
<td>$x_2 \lor x_{12}$</td>
<td></td>
</tr>
<tr>
<td>$\overline{x}<em>3 \lor \overline{x}</em>{12} \lor x_{13}$</td>
<td>$\overline{x}<em>3 \lor \overline{x}</em>{12} \lor x_{13}$</td>
<td></td>
</tr>
<tr>
<td>$x_3 \lor x_7 \lor \overline{x}_{13}$</td>
<td>$x_3 \lor x_7 \lor \overline{x}_{13}$</td>
<td></td>
</tr>
<tr>
<td>$x_8 \lor \overline{x}<em>7 \lor \overline{x}</em>{12}$</td>
<td>$x_8 \lor \overline{x}<em>7 \lor \overline{x}</em>{12}$</td>
<td></td>
</tr>
</tbody>
</table>

$x_{13}, \overline{x}_7$ appear in unary clauses
## An example of DPLL

### Formula

- $x_1 \lor x_4$
- $\overline{x}_1 \lor x_4 \lor x_{14}$
- $x_1 \lor \overline{x}_3 \lor \overline{x}_8$
- $x_1 \lor x_8 \lor x_{12}$
- $x_2 \lor x_{12}$
- $\overline{x}_3 \lor \overline{x}_{12} \lor x_{13}$
- $x_3 \lor x_7 \lor \overline{x}_{13}$
- $x_8 \lor \overline{x}_7 \lor \overline{x}_{12}$

### Simplified Formula

- $x_1 \lor x_4$
- $\overline{x}_1 \lor x_4 \lor x_{14}$
- $x_1 \lor \overline{x}_3 \lor \overline{x}_8$
- $x_1 \lor x_8 \lor x_{12}$
- $x_2 \lor x_{12}$
- $\overline{x}_3 \lor \overline{x}_{12} \lor x_{13}$
- $x_3 \lor x_7 \lor \overline{x}_{13}$
- $x_8 \lor \overline{x}_7 \lor \overline{x}_{12}$

### Partial Model

<table>
<thead>
<tr>
<th>Lev.</th>
<th>Lit.</th>
<th>Back?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\overline{x}_1$</td>
<td>(d)</td>
</tr>
<tr>
<td>+</td>
<td>$x_4$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$x_3$</td>
<td>(d)</td>
</tr>
<tr>
<td>+</td>
<td>$\overline{x}_8$</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>$x_{12}$</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>$x_{13}$</td>
<td></td>
</tr>
</tbody>
</table>

$x_7, \overline{x}_7$ appear in unary clauses
## An example of DPLL

<table>
<thead>
<tr>
<th>Formula</th>
<th>Simplified Formula</th>
<th>Partial Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 \lor x_4$</td>
<td>$x_1 \lor x_4$</td>
<td></td>
</tr>
<tr>
<td>$\overline{x_1} \lor x_4 \lor x_{14}$</td>
<td>$\overline{x_1} \lor x_4 \lor x_{14}$</td>
<td></td>
</tr>
<tr>
<td>$x_1 \lor \overline{x_3} \lor \overline{x_8}$</td>
<td>$x_1 \lor \overline{x_3} \lor \overline{x_8}$</td>
<td></td>
</tr>
<tr>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td>$x_1 \lor x_8 \lor x_{12}$</td>
<td></td>
</tr>
<tr>
<td>$x_2 \lor x_{12}$</td>
<td>$x_2 \lor x_{12}$</td>
<td></td>
</tr>
<tr>
<td>$\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$</td>
<td>$\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$</td>
<td></td>
</tr>
<tr>
<td>$x_3 \lor x_7 \lor \overline{x_{13}}$</td>
<td>$x_3 \lor x_7 \lor \overline{x_{13}}$</td>
<td></td>
</tr>
<tr>
<td>$x_8 \lor \overline{x_7} \lor \overline{x_{12}}$</td>
<td>$x_8 \lor \overline{x_7} \lor \overline{x_{12}}$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Lev. Lit. Back?</strong></td>
</tr>
<tr>
<td>1</td>
<td>$\overline{x_1}$</td>
<td>(d)</td>
</tr>
<tr>
<td>+</td>
<td>$x_4$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$x_3$</td>
<td>(d)</td>
</tr>
<tr>
<td>+</td>
<td>$\overline{x_8}$</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>$x_{12}$</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>$x_{13}$</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>$\overline{x_7}$</td>
<td></td>
</tr>
</tbody>
</table>

**Conflict! Undo everything until last decision**
An example of DPLL

<table>
<thead>
<tr>
<th>Formula</th>
<th>Simplified Formula</th>
<th>Partial Model</th>
</tr>
</thead>
</table>
| $x_1 \lor x_4$ | $x_1 \lor x_4$ | $\begin{array}{c|c}
1 & \overline{x_1} \\
+ & x_4 \\
\end{array}$
| $\overline{x_1} \lor x_4 \lor x_{14}$ | $\overline{x_1} \lor x_4 \lor x_{14}$ | (d) |
| $x_1 \lor \overline{x_3} \lor \overline{x_8}$ | $x_1 \lor \overline{x_3} \lor \overline{x_8}$ | + |
| $x_1 \lor x_8 \lor x_{12}$ | $x_1 \lor x_8 \lor x_{12}$ | $x_4$ |
| $x_2 \lor x_{12}$ | $x_2 \lor x_{12}$ | * |
| $\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$ | $\overline{x_3} \lor \overline{x_{12}} \lor x_{13}$ | $\overline{x_3}$ |
| $x_3 \lor x_7 \lor \overline{x_{13}}$ | $x_3 \lor x_7 \lor \overline{x_{13}}$ | |
| $x_8 \lor \overline{x_7} \lor \overline{x_{12}}$ | $x_8 \lor \overline{x_7} \lor \overline{x_{12}}$ | |

Now, $\overline{x_3}$ is not a decision
A very simple procedure?

Very simple to write a backtrack search but...

Where to branch?

- Mistakes at the top of the tree are dramatic!
- (almost) As many nodes where to decide than where to explore
- A perfect branching scheme is NP-Hard

Imitate how humans are solving toy examples

Cannot cope with (very) large examples!
Applications: Huge problems to come! (Literally)
Bounded Model Checking at a glance

We have a system to verify, modeled by an automaton, encoding its state transitions.

**Correctness: No bugs** A special state "error" is used in the model. The problem is about its reachability.

**Liveness: No infinite loop** Any state must be reachable from any other state, in any future.

**Notice:**
> Closely related to temporal logic;
> Before SAT, BDD were used to solve these problems.
(Bounded) Model Checking at a glance

We fix a *bound* $k$, and increment it as we need. 

- The automaton is represented by the propositional logic function $T$ that encodes the characteristics function of the reachable states.

Example (2-bits 1-adder):

\[
(a' \leftrightarrow \neg a) \land (b' \leftrightarrow a \oplus b)
\]

\[
\begin{align*}
(a' \leftrightarrow \neg a) \land (b' \leftrightarrow a \oplus b) & \\
(0, 0) \rightarrow (1, 0) \rightarrow (0, 1) \rightarrow (1, 1) \rightarrow (0, 0) \rightarrow \ldots
\end{align*}
\]
(Bounded) Model Checking at a glance

We fix a bound $k$, and increment it as we need

- The automaton is represented by the propositional logic function $T$ that encodes the characteristics function of the reachable states. Example (2-bits 1-adder): $(a' \leftrightarrow \neg a) \land (b' \leftrightarrow a \oplus b)$
- The property to check is (for instance): $a \land b$ (is the state (11) reachable?)
- The initial state is an assignment of variables at time step 0
BMC: Unrolling loops

Let us check whether the state (11) is reachable in 2 iterations

$I(s_0) = \neg a_0 \land \neg b_0$

$T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0)$

$T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1)$

$p(s_2) = a_2 \land b_2$

$p(s_0) = a_0 \land b_0$

$p(s_1) = a_1 \land b_1$

Finally, is the formula

$$(\neg a_0 \land \neg b_0) \land ((a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0)) \land ((a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1)) \land (a_2 \land b_2)$$

satisfiable?
BMC: Unrolling loops

Let us check whether the state \((11)\) is reachable in 2 iterations

\[
I(s_0) = \neg a_0 \land \neg b_0
\]

\[
T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0)
\]

\[
T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1)
\]

\[
p(s_2) = a_2 \land b_2
\]

\[
p(s_0) = a_0 \land b_0
\]

\[
p(s_1) = a_1 \land b_1
\]

Finally, is the formula

\[
(\neg a_0 \land \neg b_0) \land ((a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0)) \land ((a_2 \leftrightarrow \neg a_1) \\
\land (b_2 \leftrightarrow a_1 \oplus b_1)) \land (a_2 \land b_2)
\]
satisfiable?
BMC: Unrolling loops

Let us check whether the state (11) is reachable in 2 iterations

\[ I(s_0) = \neg a_0 \land \neg b_0 \]
\[ T(s_0, s_1) = (a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0) \]
\[ T(s_1, s_2) = (a_2 \leftrightarrow \neg a_1) \land (b_2 \leftrightarrow a_1 \oplus b_1) \]
\[ p(s_2) = a_2 \land b_2 \]
\[ p(s_0) = a_0 \land b_0 \]
\[ p(s_1) = a_1 \land b_1 \]

**Finally,** is the formula

\[
(\neg a_0 \land \neg b_0) \land ((a_1 \leftrightarrow \neg a_0) \land (b_1 \leftrightarrow a_0 \oplus b_0)) \land ((a_2 \leftrightarrow \neg a_1) \\
\land (b_2 \leftrightarrow a_1 \oplus b_1)) \land (a_2 \land b_2)
\]
satisfiable?
Huge problems?
Generated by the tool: satgraf http://satbench.uwaterloo.ca/site/satgraf
Huge problems?
Generated by the tool: satgraf http://satbench.uwaterloo.ca/site/satgraf
Huge problems?
Generated by the tool: satgraf http://satbench.uwaterloo.ca/site/satgraf
Huge problems?
Generated by the tool: satgraf http://satbench.uwaterloo.ca/site/satgraf
Huge problems?
Generated by the tool : satgraf http://satbench.uwaterloo.ca/site/satgraf
Unbounded Model Checking

\[ I \land T_1 \land T_2 \land \ldots \land T_k \land BUG_k \]

How to ensure that \( BUG \) is unreachable?

**Idea:** find an invariant \( Inv \) s.t. \( BUG \) is not reachable in \( k > 0 \) steps

- \( Inv \) characterizes an over approximation of the reachable states in \( j \) steps:

\[ I \land T_1 \land \cdots \land T_j \rightarrow Inv \]

- \( Inv \) is an inductive property:

\[ Inv \land T_1 \rightarrow Inv_1 \]

- \( BUG \) is not reachable from \( Inv \) in \( k \) steps:

\[ Inv \land T_1 \land T_2 \land \ldots \land T_k \land BUG_k \equiv \bot \]

- Incremental SAT Solving / Proof Analysis
The "biggest proof" in the world

Boolean Pythagorean triples problem
Is it possible to colorize the $n$ integers $\leq n$ in two colors s.t. no triplet $(a, b, c)$ is $a^2 + b^2 = c^2$ monochromatic?
The "biggest proof" in the world

No solution for \( n=7825 \)

- Open question since 20 years
- \( 10^{2300} \) possible candidates
- SAT encoding (trivial):
  Add the clauses \( i \lor j \lor k \) and \( \neg i \lor \neg j \lor \neg k \) for all \( i, j, k \) that are Pythagorean triplets.
- Original problem splitted in 1,000,000 subproblems
- 800 CPUs

Proof is 200Tb long (Glucose’s output)

- In practice the proof is not really kept
CDCL solvers, in a few words
1999, on the way to the revolution

Huge problems are coming from the real-world: Planning & Bounded Model Checking

- Planning as Satisfiability. [Kautz and Selman, 92]
- Symbolic Model Checking using SAT procedures instead of BDDs. [Biere & al. 99]
- SAT solvers can’t cope with those huge formulas without specialized data structures

DPLL extinction...

- GRASP: **learning clauses** in SAT solvers
- DLIS: **very simple heuristic**
- SATO: **lazy data structure** to detect unary clauses

**Algorithms ingredients for the upcoming revolution**
BMC, GRASP, DLIS, SATO
Data structures are stronger than algorithms!

Until now:

> Heuristics were used to *mimic* what a human will do (when picking a variable to branch on)

> Data structures were implemented to support algorithms

**Seminal paper**

« *Chaff: Engineering an Efficient SAT Solver* » [Moskewicz & al. ’01]

“Simply” optimize known algorithms

**Highest priority**: BCP (Boolean Constraint Propagation)
We have to cope with this lazy data structure

The current state of the formula is unknown!

> How many reduced clauses? how many satisfied clauses?
> Some variables may be pure?

**Only one guarantee:** all unary and empty clauses are detected

**Find a model?** all the variables are assigned without any conflict

**Heuristics:** How to choose a variable to branch on? We are blind!
> We need to use the past, not the current state of the formula
> The heuristics will be heavily related to the clause learning mechanism
Ingredients of an efficient SAT solver

- Preprocessing (and inprocessing)
  - Restarting
  - Branching
  - Conflict Analysis
  - Clause Database Cleaning
CDCL principles at a glance

Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \]
CDCL principles at a glance
Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x}_3 \lor \overline{x}_7 \lor x_{13} \]
\[ \phi_6 = \overline{x}_3 \lor \overline{x}_7 \lor \overline{x}_{13} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x}_7 \lor \overline{x}_9 \]
CDCL principles at a glance

Decisions – Propagations

\[
\begin{align*}
\phi_1 &= x_1 \lor x_4 \\
\phi_2 &= x_1 \lor \overline{x_3} \lor \overline{x_8} \\
\phi_3 &= x_1 \lor x_8 \lor x_{12} \\
\phi_4 &= x_2 \lor x_{11} \\
\phi_5 &= \overline{x_3} \lor \overline{x_7} \lor x_{13} \\
\phi_6 &= \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \\
\phi_7 &= x_8 \lor \overline{x_7} \lor \overline{x_9}
\end{align*}
\]
CDCL principles at a glance

Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \]
CDCL principles at a glance
Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \]
CDCL principles at a glance
Decisions – Propagations

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x}_3 \lor \overline{x}_7 \lor x_{13} \]
\[ \phi_6 = \overline{x}_3 \lor \overline{x}_7 \lor \overline{x}_{13} \lor x_9 \]
\[ \phi_7 = x_8 \lor x_7 \lor \overline{x}_9 \]
CDCL principles at a glance

Decisions – Propagations

\[\phi_1 = x_1 \lor x_4\]
\[\phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8}\]
\[\phi_3 = x_1 \lor x_8 \lor x_{12}\]
\[\phi_4 = x_2 \lor x_{11}\]
\[\phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13}\]
\[\phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9\]
\[\phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9}\]
CDCL principles at a glance
Decisions – Propagations

ϕ₁ = x₁ ∨ x₄
ϕ₂ = x₁ ∨ x₃ ∨ x₈
ϕ₃ = x₁ ∨ x₈ ∨ x₁₂
ϕ₄ = x₂ ∨ x₁₁
ϕ₅ = x₃ ∨ x₇ ∨ x₁₃
ϕ₆ = x₃ ∨ x₇ ∨ x₁₃ ∨ x₉
ϕ₇ = x₈ ∨ x₇ ∨ x₉
CDCL principles at a glance

Conflict Analysis

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \]
CDCL principles at a glance

Conflict Analysis

\[ \beta_1 = res(x_9, \phi_7, \phi_6) = \overline{x}_3 \lor x_8 \lor \overline{x}_7 \lor \overline{x}_{13} \]

\[
\begin{align*}
\phi_1 &= x_1 \lor x_4 \\
\phi_2 &= x_1 \lor \overline{x}_3 \lor \overline{x}_8 \\
\phi_3 &= x_1 \lor x_8 \lor x_{12} \\
\phi_4 &= x_2 \lor x_{11} \\
\phi_5 &= \overline{x}_3 \lor \overline{x}_7 \lor x_{13} \\
\phi_6 &= \overline{x}_3 \lor \overline{x}_7 \lor \overline{x}_{13} \lor x_9 \\
\phi_7 &= x_8 \lor \overline{x}_7 \lor \overline{x}_9
\end{align*}
\]
CDCL principles at a glance

Conflict Analysis

\[ \beta_1 = res(x_9, \phi_7, \phi_6) = \overline{x_3} \lor x_8 \lor \overline{x_7} \lor \overline{x_{13}} \]
\[ \beta = res(x_{13}, \beta_1, \phi_5) = \overline{x_3} \lor x_8 \lor \overline{x_7} \]

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8} \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13} \]
\[ \phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9 \]
\[ \phi_7 = x_8 \lor \overline{x_7} \lor \overline{x_9} \]
CDCL principles at a glance

Conflict Analysis

\[ \beta_1 = res(x_9, \phi_7, \phi_6) = \overline{x}_3 \lor x_8 \lor \overline{x}_7 \lor x_{13} \]

\[ \beta = res(x_{13}, \beta_1, \phi_5) = \overline{x}_3 \lor x_8 \lor \overline{x}_{7} \]

- Stops as soon as the resolvant has a unique literal from the last decision level (FUIP).
- \( \beta \) is added to the clauses databases (ensure a systematic search).
CDCL principles at a glance
Non Chronological Backtrackings

\[ \phi_1 = x_1 \lor x_4 \]
\[ \phi_2 = x_1 \lor \overline{x}_3 \lor \overline{x}_8 \]
\[ \phi_3 = x_1 \lor x_8 \lor x_{12} \]
\[ \phi_4 = x_2 \lor x_{11} \]
\[ \phi_5 = \overline{x}_3 \lor x_7 \lor x_{13} \]
\[ \phi_6 = \overline{x}_3 \lor x_7 \lor \overline{x}_{13} \lor x_9 \]
\[ \phi_7 = \overline{x}_8 \lor \overline{x}_7 \lor x_9 \]

\[ \beta = \overline{x}_3 \lor x_8 \lor \overline{x}_7 \]
CDCL principles at a glance
Non Chronological Backtrackings

\( \phi_1 = x_1 \vee x_4 \)
\( \phi_2 = x_1 \vee \overline{x_3} \vee \overline{x_8} \)
\( \phi_3 = x_1 \vee x_8 \vee x_{12} \)
\( \phi_4 = x_2 \vee x_{11} \)
\( \phi_5 = \overline{x_3} \vee x_7 \vee x_{13} \)
\( \phi_6 = \overline{x_3} \vee x_7 \vee \overline{x_{13}} \vee x_9 \)
\( \phi_7 = \overline{x_8} \vee \overline{x_7} \vee x_9 \)

\( \beta = \overline{x_3} \vee x_8 \vee \overline{x_7} \)
CDCL principles at a glance
Non Chronological Backtrackings

$\phi_1 = x_1 \lor x_4$
$\phi_2 = x_1 \lor \overline{x_3} \lor \overline{x_8}$
$\phi_3 = x_1 \lor x_8 \lor x_{12}$
$\phi_4 = x_2 \lor x_{11}$
$\phi_5 = \overline{x_3} \lor \overline{x_7} \lor x_{13}$
$\phi_6 = \overline{x_3} \lor \overline{x_7} \lor \overline{x_{13}} \lor x_9$
$\phi_7 = \overline{x_8} \lor \overline{x_7} \lor x_9$

$\beta = \overline{x_3} \lor x_8 \lor \overline{x_7}$
What about your problems?
How to use a SAT solver?

**General Principles**: don’t be afraid of inserting a lot of variables

**Sudoku**: 999 variables, >12018 clauses, always solved in 0.0s

**Use actual tools for translation in SAT**

```python
from pysat.solvers import Glucose3

g = Glucose3()
g.add_clause([-1, 2])
g.add_clause([-2, 3])
print(g.solve())
print(g.get_model())
...
True
[−1, −2, −3]
```

https://github.com/pysathq/pysat
Use Sugar before Glucose

**Sugar** is a constraints-to-SAT CSP solver using *order encoding*.

All the comparisons $x \leq a$ are encoded by a single variable (for each $x, a$)

Sugar won a number of awards in the constraints competitions

You can use it to translate your problem from a higher language to SAT

... 
(int x_5_5 1 25)
(alldifferent x_1_1 x_1_2 ... x_5_5)
(or (= x_1_1 25) (= (+ x_1_1 1) x_2_3) (= (+ x_1_1 1) x_3_2))
...

[bach.istc.kobe-u.ac.jp/sugar/](http://bach.istc.kobe-u.ac.jp/sugar/)
And beyond NP?

**SAT solvers can be used in problems beyond NP:**

- Optimization
- Minimal Explanation
- Counting
- ...

**Incremental SAT Solving is a powerful technique**

(and made easy with PySAT)

Successively call SAT solvers by

- Activating new clauses
- Deactivating some clauses
CDCL solvers are complex systems

We have a lot of open problems around these questions:

"Understand what we have implemented"

It’s ok if we don’t fully “understand” our code

- Very fast and unpredictable
- Work well on real-world instances, but how to define such a structure?
- All components are tightly connected, side effects are everywhere
- There is no “one-simple reason” explaining their performance (supposition)
- At least we know that we don’t know

Idea behind glucose

A real experimental study of CDCL solvers
CDCL solvers are complex systems – Illustration

Example of a real conflict analysis:

> Many resolutions at each conflict
> Very reactive VSIDS (1/10s lifetime)

But: A clear structure behind!

A part of the research in SAT is hunting for the structure behind CDCL mechanisms
How to understand this (trivial) proof?

- Less than 70 000 conflicts
- Solved in a few seconds
- A very dense proof
- Hard to understand
Use this technology!